Discovery, Vol. 52, No. 246, June 1, 2016

RESEARCH

ISSN 2278–5469 EISSN 2278–5450

Discovery

Optimal Logic Circuit Design for Reversible Quantum Computation Based on Excess-3 Code

Jeong Ryeol Choi[☼]

Professor, Department of Radiologic Technology, Daegu Health College, Yeongsong-ro 15, Buk-gu, Daegu 41453, Republic of Korea

[☼]Corresponding author: Department of Radiologic Technology, Daegu Health College, Yeongsong-ro 15, Buk-gu, Daegu 41453, Republic of Korea, e-mail: choiardor@hanmail.net

Article History
Received: 05 April 2016
Accepted: 17 May 2016
Published: 1 June 2016

Citation
Jeong Ryeol Choi. Optimal Logic Circuit Design for Reversible Quantum Computation Based on Excess-3 Code. *Discovery*, 2016, 52(246), 1483-1493

ABSTRACT

If the classical computers would be replaced by quantum computers in the future, the algorithms for reversible quantum computations are necessary. Such reversible computations resolve the problem of heat dissipation in computers, that is an essential weak point in the conventional computing process, and enable the most powerful and fastest computing operation. The method of reversible quantum computation using Excess-3 code is addressed in this work. Excess-3 code provides a convenient way for quantum implementation of decimal arithmetic algorithms in computer. A reversible decimal arithmetic circuit based on Excess-3 code is constructed with the use of the Full and Half Adders, according to the flowchart for the arithmetic process. Further, by improving it, an optimized quantum circuit for Excess-3 Adder, which performs more efficient operations, is designed.

Keywords: Quantum Computing, Quantum Circuit, Excess-3 Code, Adder, Arithmetic Operation

Abbreviations: VLSI - very large scale integrated, BCD - binary coded decimal

1. INTRODUCTION

Up until recently, common trends for designing VLSI (very large scale integrated) circuits were to pack more and more logic components in smaller and smaller volumes and, simultaneously, to increase the speed of logic operations.

Moore's law [1] predicts an exponential growth of the capacity and complexity of integrated circuits. More precisely, it predicts that the processing power becomes double every 18 months. Although this law has continued to hold true up to recently, it would prove to be unsustainable in the near future due to the limits of reducing the size of electronic components. Hence, Moore's law may inevitably be modified or discarded, while the necessity to devise high-speed computers that overcome the limits of the conventional computer continuously increases. Moreover, conventional computers release more and more heat as the circuits become increasingly complicated than ever before, because their logic operations are basically irreversible processes. According to the Landauer's principle [2], each bit of lost information in irreversible logic computation brings about some amount of energy dissipation. The logical process of most conventional gates used in digital design, such as OR, EXOR, and NAND, is basically irreversible. The only reversible gate traditionally used is NOT gate. The released energy for a single bit lost information is kT ln2, where k is the Boltzmann constant and T is the absolute temperature at which the operation is carried out.

The best solution for these problems is to realize quantum computation using well-defined reversible logic gates [3]. Reversible computation does not bring about energy dissipation relevant to information loss, since the informations are not erased during the computation process. The mechanism for operating reversible logic circuits is essentially different from that of conventional logic circuits [4-6]. Bernnett reexamined Landauer's principle and showed that it is possible to design a reversible general-purpose Turing machine that enables one to reset its scratchpad into the initial state before ending the computation process. Recent emergent reversible quantum logic circuits provide a promising paradigm that plays a crucial role for quantum computation, which will be applied to next generation technologies, such as quantum dot cellular automata [7], programmable logic circuit array [8], optical computing [9], and quantum-optical networks [10]. As is well known, the laws of classical physics which has been the principal theory for manufacturing conventional computers no longer hold in quantum computers. Instead, the quantum effects would become increasingly important when the components of a circuit are ever to operate at the atomic level [11].

According to this trend, we will show in this work how to design reversible quantum logic circuits that perform quantum computations by means of a special code. Although binary code is the most basic one among many available codes for counting numbers in computer hardware systems, it is sometimes inadequate to be used in numerical operations in computers. For example, errors appear inevitably when converting decimal codes into binary ones, because there is no one-to-one correspondence between decimal and binary codes [12]. Hence, decimal codes, such as BCD (binary coded decimal) code [13], are necessary for error-free computations. This is the reason why decimal arithmetic provides promising uses in the commercial and financial applications of everyday computations.

There is a more convenient code for executing decimal arithmetic operations, which is Excess-3 code [14]. The expression of any numerical character done through Excess-3 code is composed of 4 bits like BCD code, but each number is added by 3 from the corresponding BCD number (see table 1). Excess-3 code has the property of self complement which guarantees the merit of the code in decimal arithmetic. So to speak, 1's complement of an Excess-3 number is identical to 9's complement of the corresponding decimal number represented by Excess-3 code. For instance, 1's complement of 0110 (3) is 1001 (6) which is the 9's complement of 3. This peculiar property of Excess-3 code is sometimes useful when designing the implementation for the arithmetic operation in Adders [14].

Recently, the possibility for realizing the construction of quantum logic circuits using reversible Adders has been reported [5-7, 12, 15]. Stimulated by this in part, a quantum logic circuit necessary for any arithmetic operations on quantum computers will be designed using Excess-3 code in this paper. Further, this basic logic circuit will be improved via reducing the garbage lines of quantum logic operations, so that it can be operated more efficiently. In order to achieve optimal quantum computations by exploiting minimized quantum resources, it will be most important to simplify the computing process so far as the same results are given.

Table 1 Comparison between Binary, BCD, and Excess-3 codes

No.	Binary code	BCD Code	Excess-3 code
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	0101	1000
6	0110	0110	1001
7	0111	0111	1010
8	1000	1000	1011
9	1001	1001	1100
10	1010	.	.
11	1011	.	.
12	1100	.	.
13	1101	.	.
14	1110	.	.
15	1111	.	.

2. METHODS

A. Summary of methods

We first introduce Half and Full Adders that are constructed by means of reversible logic gates such as controlled-NOT gate and Toffoli gate. A flowchart for the process of decimal arithmetic computations based on Excess-3 code is drawn up with Half and Full Adders using the merit of Excess-3 code, which is its self complement property. According to the flowchart, a reversible Excess-3 Adder is designed using reversible logic gates. Further, the primitive Excess-3 Adder is improved so that it can perform optimal quantum computations with reduced garbage outputs.

B. Methods of arithmetic operations with examples

To design reversible Excess-3 Adders, it is necessary to see the numerical process of arithmetic operations with Excess-3 code [15]. The novel advantage of Excess-3 code mentioned earlier can be conveniently exploited in decimal arithmetic. Recall that the 1's complement of an Excess-3 number is the same as the 9's complement of the corresponding decimal number represented upon Excess-3 basis. For better understanding of this, we now explain the arithmetic operation with several examples as follows.

- **Example 1:** Arithmetic operations which do not produce a carry.

Let us evaluate 3+5=8 using Excess-3 code. In this case, each step is as follows:
1. First, add 0110 (3) and 1000 (5). Then, it gives 1110.
2. Because a carry did not occur in the first step, it is necessary to subtract 0011 from 1110. Thus, the result becomes 1011 (8).

Notice that there is another method equivalent to this computation as given below:
1. The first step is the same as the first one in the above process, leading to the result, 1110.
2. Add 1100 (that is the 1's complement of 0011) into 1110. This operation leads to 1 1010.
3. Add the carry 0001, which occurred in the second step, into 1010. Then, the final result is 1011 (8).

As you can see, the second method gives exactly the same result as the first one. The merit of the second method is that the arithmetic process can be fulfilled without using any Subtractors, i.e., the overall process can be implemented by means of using only Adders. We will design the Excess-3 Adder using the second method. You can see from the flowchart (Fig. 2) that the second method is more convenient than the first one for designing an Excess-3 Adder.

On the other hand, one should add 0011, if the number produced by adding arbitrary two numbers together is greater than or equal to decimal 10. To illustrate this, let us see the following second example.

- **Example 2:** Arithmetic operations which produce a carry.

Now we consider 7+9 for another example of computation with Excess-3 code. The result of this process is 6 with a carry 1. The steps for this evaluation are as follows:

1. First, add 1010 (7) and 1100 (9). Then, it will give 1 0110.
2. You can see that the carry is produced in the first step. In this case, it is necessary to add 0011 into 0110. Then the result will be 1001 (6).
3. Simultaneously, the carry occurred in the first step should be sent to the next rank of 4-digit Adder where the subsequent arithmetic operation will be carried out. Hence, the eventual result becomes 1001 (6) with the carry 1, as expected.

3. ADDERS AND QUANTUM COMPUTATION

Among various arithmetic circuits, Adder is the most basic block in a computing system. Other arithmetic operations required in a number of digital signal processes, such as subtraction, multiplication, and division, can be fulfilled by the application of Adders [7]. This implies that an efficient design of an Adder can be of great assistance in the whole process of quantum computing. While a Half Adder is composed of two inputs, a Full Adder has three inputs. The diagrams for Half and Full Adders are illustrated in Fig. 1. The two inputs in Full Adder (Fig. 1(b)), designated as A and B, are the two customary qubits to be added and the additional input qubit C_i is for the carry produced from the former step.

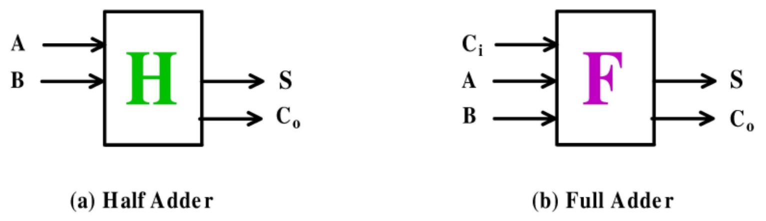

Figure 1 Diagrams for Adders

In recent years, decimal arithmetic received considerable interest, not only as an application in commercial and financial computing tasks, but also in other diverse internet-based applications which require accurate numerical results. This trend is originated from the fact that the operations of decimal arithmetic do not tolerate errors, while inevitable errors appear whenever converting any binary arithmetic results into decimal numbers [12] or vice versa. It is well known that some of decimal numbers such as 0.230 cannot be represented by binary numbers. Thus, the data that include these numbers should be stored in decimal format for accuracy. One of the most widely used codes for representing decimal numeric data is BCD code. BCD code is used to record each decimal digit of a number by a 4-bit binary value. The BCD numbers for 0 and 9 are 0000 and 1001, respectively: this means that the binary numbers 1010 to 1111 are invalid in BCD code (see table 1).

Another code that is convenient when recording decimal number is Excess-3 code [12], as mentioned in the introductory part. All numbers in Excess-3 code are represented by adding 3 (0011 in BCD code) to the corresponding BCD numbers (see table 1). For instance, to express a number, 236, in terms of Excess-3 code, one simply needs to encode each of the decimal digits to be 0101 0110 1001.

When adding two arbitrary numbers together, represented by Excess-3 code, we should be careful since the result is not an exact Excess-3 number. For example, if you naively add together two Excess-3 code numbers, 0100 (1) and 0101 (2), the answer is 1001 (6) instead of the precise number 0110 (3). In order to correct this problem, one should subtract 0011 after adding each digit if the result is less than 10 in decimal number. For further and better understanding of Excess-3 arithmetic, see the two examples given in Sec. 2. BCD code can be easily converted to Excess-3 code and vice versa by a simple specific circuit [12].

As a preliminary step to design an Excess-3 Adder, one should know the flowchart [15] of the whole process of numerical operation. You can see such chart built from a quantum reversible point of view in Fig. 2. We will design an Excess-3 Adder according to this chart.

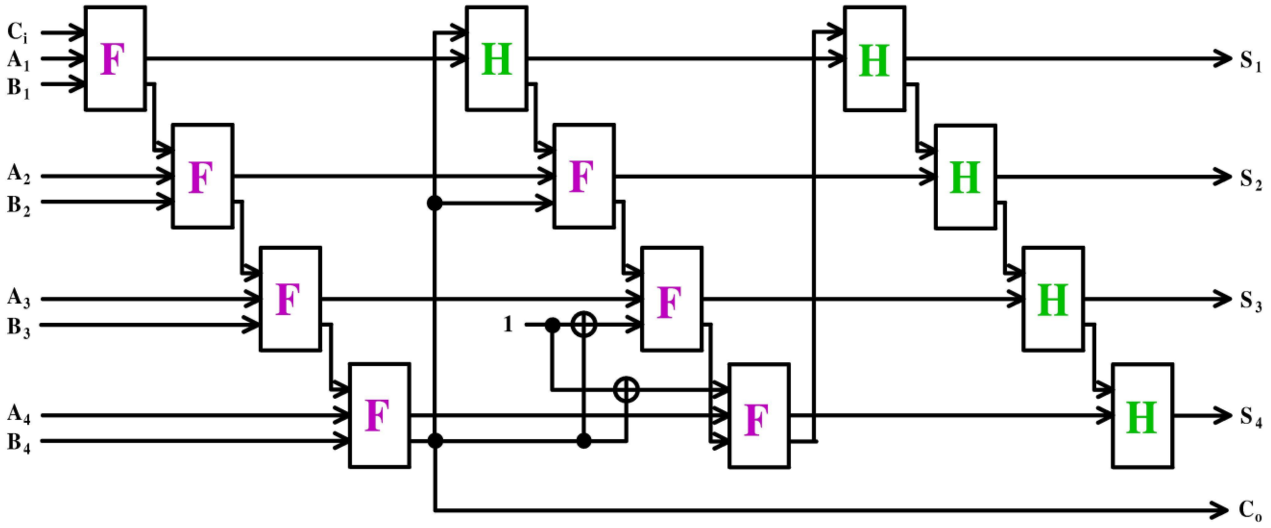

Figure 2 Flowchart for the process of quantum computing with Excess-3 code using Half and Full Adders. We assume that the two numerical numbers represented in terms of Excess-3 code are $A_4A_3A_2A_1$ and $B_4B_3B_2B_1$. The addition of $A_4A_3A_2A_1$, $B_4B_3B_2B_1$, and the carry (C_i), flowed in this step from the previous step, results in $S_4S_3S_2S_1$ with the new carry C_o. If there is a carry occurred in this step, it will move to the next step and be added to the new operation of computations.

4. LOGIC CIRCUIT DESIGN

The issue of designing reversible logic circuits is firstly addressed by Toffoli [16], as far as we know. He further strengthened his ideas for this via one of his pioneering works [17] on conservative process of logic. Afterwards, there have been various efforts for constructing reversible sequential circuits by many researchers [5, 6, 8, 12, 18-20].

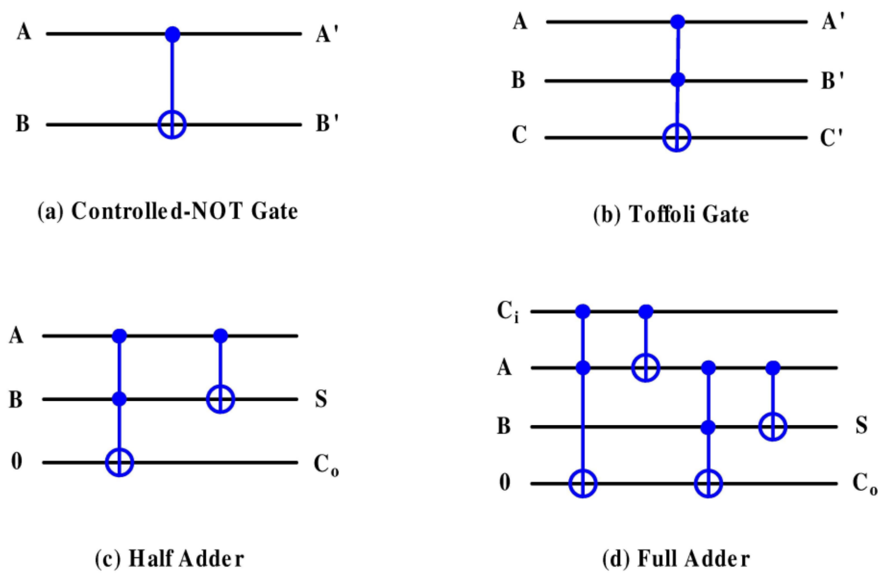

Figure 3 Reversible Adders

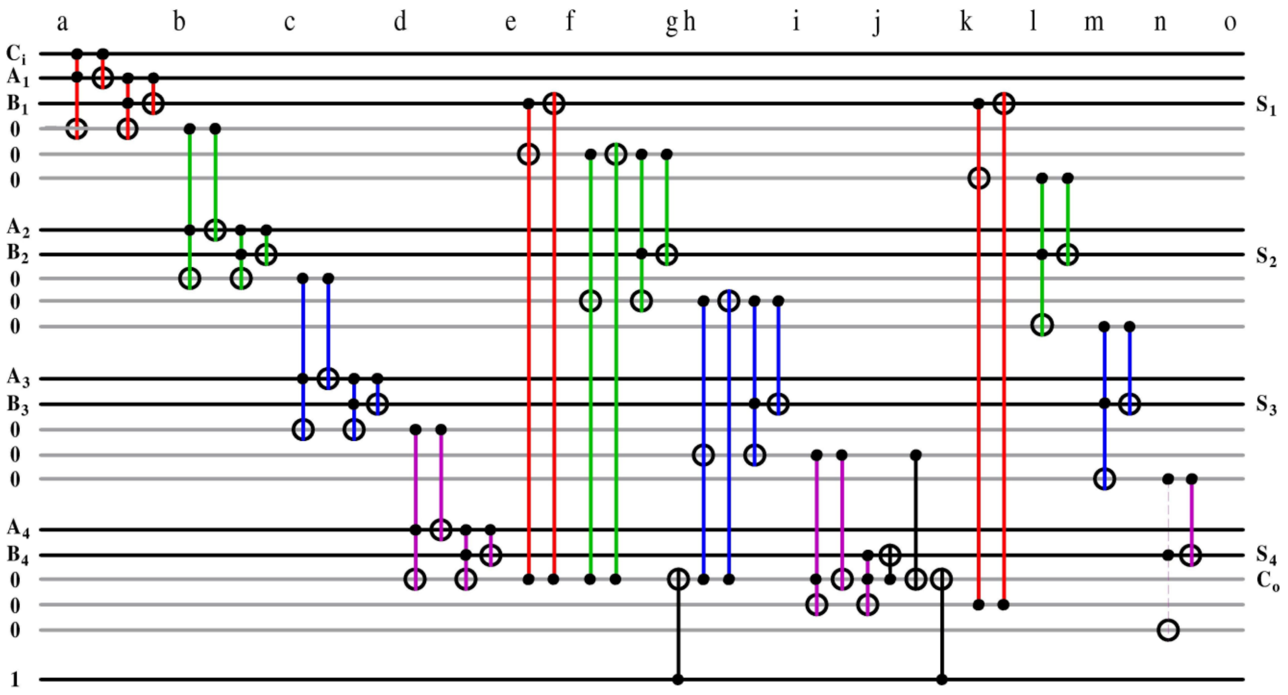

Figure 4 Excess-3 Adder

We now see how to construct reversible Adders using elementary quantum gates. Elementary quantum gates such as controlled-NOT gate and Toffoli gate are basically reversible. Let us do an overview of these gates at first. If we denote inputs of a controlled-NOT gate as A (the control qubit) and B (the target qubit), the outputs A' and B' are represented as

$$A' = A, \qquad B' = A \oplus B.$$

This means that the second qubit of the Controlled-NOT gate flips, if and only if the first qubit is 1.

Toffoli gate has three inputs and the same number of outputs. The outputs of this gate are given by

$$A' = A, \qquad B' = B, \qquad C' = C \oplus (A \cdot B).$$

When the first two qubits are set to be 1 simultaneously, the third qubit flips. The symbols of the controlled-NOT gate and the Toffoli gate are represented in Figs. 3 (a) and (b), respectively. Other reversible quantum gates are also investigated in the literature [4, 21].

Quantum circuits are diagrams that represent time evolution of the process of quantum computation executed using fundamental gates. A reversible quantum circuit is usually represented by placing a series of necessary quantum gates on a number of parallel lines. There is a one-to-one correspondence between the input and the output qubit values, because the number of outputs in such circuits is exactly the same as that of inputs. For this reason, unique output results are generated from each input information and the process can be reversible. Well established synthetic methods of this style are useful for reducing the size of circuits with minimization of garbage outputs, while the quantum cost, which is the number of primitive gates in a circuit, and garbage outputs are the primary concerns in quantum logic circuit design [22]. If there is an input of the gate which is not used as the input in the subsequent gates, we call it a garbage output. Garbage outputs are of course irrelevant to primary outputs.

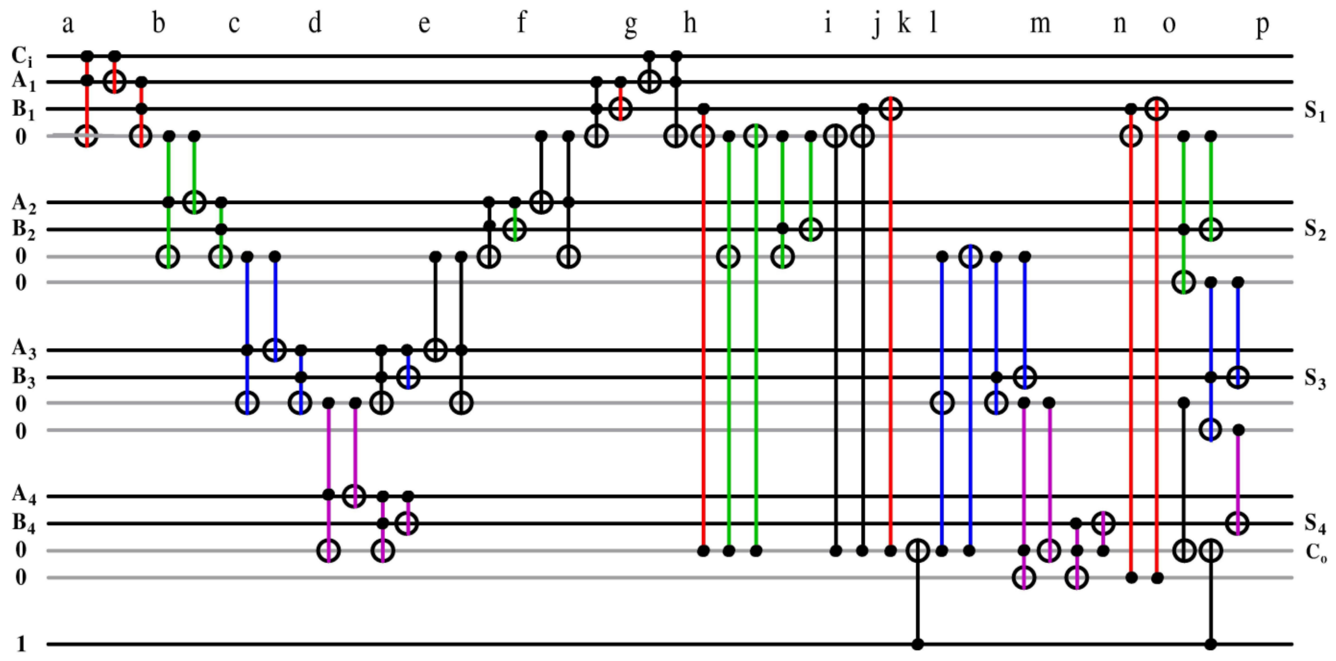

Figure 5 Improved Excess-3 Adder

One can make reversible Half and Full Adders [23, 24] as a combination of controlled-NOT gates and Toffoli gates as shown in Figs. 3(c) and (d), respectively. These Adders will be used to design logic circuits for quantum computation. Reversible quantum logic circuits keep overall information during the operation process, because the computations are carried out through reversible gates. Figure 4 is an Excess-3 Adder designed on the basis of the flowchart using Half and Full Adders discussed previously. A shortcoming of this Adder circuit is that there are too many input lines. If a system is complicated, it is more likely to incur errors in the process of computing. As well as the usual system errors, other errors induced by decoherence that takes place in the system should be removed. It is well known that decoherence leads a quantum state to be transferred to a classical state in which the concept of quantum computing is meaningless. Hence, the improvement of circuit-fabrication design, in addition to enhancing the degree of integration, is necessary for proper computation. According to this, we optimize the circuit by simplifying the operation process. Figure 5 shows an improved circuit of Excess-3 Adder. In fact, this is the central work of this research. The naive Excess-3 Adder of Fig. 4 has 22 operation lines, but the improved one given in Fig. 5 has 17 operation lines. Thus, we removed 5 garbage lines. One can easily check that the inputs of the arbitrary Excess-3 numbers ($A_4A_3A_2A_1$ and $B_4B_3B_2B_1$) in Fig. 5 give correct results ($S_4S_3S_2S_1$) that are actually identical to those obtainable from the circuit in Fig. 4 through the same inputs.

Table 2 Values of qubits at each time represented in Fig. 4. The chosen excess-3 input values are ($A_4A_3A_2A_1$, $B_4B_3B_2B_1$) = (0110, 1000) which are the same as the case of the first example in Sec. 2.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	
C_i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
B_1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	S_1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
B_2	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	S_2
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	

	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
B_3	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	S_3
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
B_4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	S_4
	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	C_o
	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

There are 12 zero input lines in Fig. 5, which are represented with gray colour in order to distinguish them from other lines. In Fig. 4, the number of zero input lines is reduced to 7. The roles of removed zero input lines are undertaken by other gray lines. For example, the roles of the second and the third zero input lines in Fig. 4 are undertaken by the first zero input line in Fig. 5. In order to reuse remained gray lines in Fig. 5 for such purpose, the values of them are reset to zero at some necessary moments. For example, the first gray line in Fig. 5 is reset to zero at the moments h and o. One can easily confirm that the value of the first line in Fig. 5 is always zero at the moments h and o regardless of the input values $A_4A_3A_2A_1$ and $B_4B_3B_2B_1$.

Table 3 The same as table 2, but the chosen excess-3 input values are ($A_4A_3A_2A_1$, $B_4B_3B_2B_1$) = (1010, 1100) which are the same as the case of the second example in Sec. 2.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	
C_i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
B_1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	S_1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
B_2	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	S_2
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
B_3	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	S_3
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
B_4	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	S_4
	0	0	0	0	1	1	1	0	0	1	1	1	1	1	1	C_o
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

For clarification of the computing results, the time evolutions of qubit values for the circuit given in Fig. 4 are represented in tables 2 and 3, and those given in Fig. 5 are represented in tables 4 and 5, for two particular Excess-3 input values. From these tables, we can confirm the validity of the improved circuit of Excess-3 Adder.

5. CONCLUSION

Logic circuit designs for reversible quantum computations using Excess-3 code are investigated. Excess-3 code is advantageous for constructing decimal Adders and for simplifying their computation process. The method to perform decimal arithmetic with Excess-3 code is addressed in detail. The flowchart for the operating process of an Excess-3 Adder is represented in Fig. 2. By rigorously following this chart, the circuit of an Excess-3 Adder is designed, as can be seen from Fig. 4.

Although the circuit in Fig. 4 is valid, more optimally designed circuits with reduced size are preferable, if we think that the possibility for realizing error-free quantum computations is a very subtle problem due to the appearance of decoherence especially when the circuit is complicated. The criterion for excellent circuit design is that it should be suitable for obtaining a robust implementation with reduced overall running time for an algorithm. Robust quantum computation can be achieved if quantum coherence is sustained, at least to some extent, during the entire process of computing. Indeed, the decoherence phenomenon is a serious obstacle to the exploitation of practical quantum computers. In actual circuits, there are errors caused by decoherence during the computation [25], as well as usual system errors or experimentally aroused errors. The effects of decoherence can be reduced by simplifying the operation process via minimizing the computing resources. Considering this, we have consequently improved the circuit by reducing the lines of quantum operations (see Fig. 5). We can confirm that the arithmetic qubit results of this improved Excess-3 Adder are exactly the same as those of the primitive Excess-3 Adder, by comparing the resulting values $S_4S_3S_2S_1$ given in tables 4 and 5 with those given in tables 2 and 3. The new circuit given in Fig. 5 is suitable for performing more efficient operations. The number of garbage outputs is fairly reduced in this new design. Optimum or near optimum results can be obtained by this scheme in nanotechnology based computing systems.

Table 4 Values of qubits at each time represented in Fig. 5 which corresponds to the improved Excess-3 Adder. The chosen excess-3 input values are ($A_4A_3A_2A_1$, $B_4B_3B_2B_1$) = (0110, 1000) which are the same as the case of the first example in Sec. 2.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	
C_i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
B_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	S_1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
B_2	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	S_2
	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
B_3	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	S_3
	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
B_4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	S_4
	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	C_o
	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

In spite of the merit of decimal arithmetic for error-free computations, as mentioned in the text, the software implementation of decimal arithmetic brings about tardiness in the processing speed, which is typically 100 to 1000 times slower than the binary arithmetic [22]. Hence, the necessity of simplifying quantum operations in decimal arithmetic is a crucial factor. The reversible Adder designed here can be used as a basic resource for building higher order quantum arithmetic circuits which are expected to support the realization of quantum computers.

Table 5 The same as table 4, but the chosen excess-3 input values are ($A_4A_3A_2A_1$, $B_4B_3B_2B_1$) = (1010, 1100) which are the same as the case of the second example in Sec. 2.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	
C_i	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
B_1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	S_1
	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
A_2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
B_2	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	S_2
	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
B_3	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	S_3
	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A_4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
B_4	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	S_4
	0	0	0	0	1	1	1	1	1	1	1	0	0	1	1	1	C_o
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

DISCLOSURE STATEMENT

This research was supported by the Basic Science Research Program of the year 2015 through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (Grant No.: 2013R1A1A2062907).

REFERENCES

1. G. E. Moore, Cramming more components onto integrated circuits. *Proc. IEEE*, vol. 86, no. 1, 82-85 (1998).
2. R. Landauer, Irreversibility and heat generation in the computing process. *IBM J. Res. Develop.*, vol. 5, no. 3, 183-191 (1961).
3. C. H. Bennet, Logical reversibility of computation. *IBM J. Res. Develop.*, vol. 17, no. 6, 525-532 (1973).
4. M. Nielsen, I. Chuang, *Quantum Computation and Quantum Information* (New Delhi, Cambridge University Press, 2002).
5. V. K. Panchal, V. H. Nayak, Analysis of multiplier circuit using reversible logic. *Int. J. Innovative Res. Sci. Technol.*, vol. 1, no. 6, 2349-6010 (2014).
6. Md. S. A. Mamun, I. Mandal, Md. Hasanuzzaman, Efficient design of reversible sequential circuit. *IOSR J. Computer Eng.*, vol. 5, no. 6, 42-47 (2012).
7. B. Sen, A. Rajoria, B. K. Sikdar, Design of efficient Full Adder in quantum-dot cellular automata. *Sci. World J.*, vol. 2013, 250802 (2013).
8. D. Sarvani, Y. Syamala, Y. Ratna Babu, Realization of reversilbe Full Adder & reversible Full Subtractor using RPLA. *Int. J. Eng. Res. Appl.*, ASPC, 56-59 (2012).
9. S. Kaur, R. S. Kaler, All optical integrated full adder-subtractor and demultiplexer using SOA-based Mach-Zehnder interferometer. *Int. J. Eng. Sci. Technol.*, vol. 4, no. 1, 303-310 (2012).
10. K. Stannigel, P. Rabl, P. Zoller, Driven-dissipative preparation of entangled states in cascaded quantum-optical networks. *New J. Phys.*, vol. 14, no. 6, 063014 (2012).
11. J. R. Choi, B. J. Choi, H. D. Kim, Displacing, squeezing, and time evolution of quantum states for nanoelectronic circuits. *Nanoscale Res. Lett.*, vol. 8, no. 1, 30 (2013).
12. H. Thapliyal, H. R. Arabnia, R. Bajpai, K. K. Sharma, Partial reversible gates(PRG) for reversible BCD arithmetic. Proceedings of the 2007 International Conference on Computer Design (CDES'07), Las Vegas, U.S.A, June 2007, pp. 90-91 (CSREA Press).
13. H. Pelka, Computation with BCD (Binary-Coded-Decimal) numbers. *Elektroniker*, vol. 15, el14-el20 (1976).
14. V. V. Blatov, A. A. Chudov, Binary-decimal adder-subtractors. *Instrum. Exp. Tech.*, vol. 21, no. 5 pt 1, 1260-1264 (1978).

15. K. H. Yeon, J. R. Choi, D. Kim, M.-S. Kim, M. Maamache, Reversible quantum computation using Excess-3 code. *AIP Conf. Proc.*, vol. 1444, 310-313 (2012).
16. T. Toffoli, Reversible computing. Tech memo MIT/LCS/TM-151, MIT Lab for Computer Science (1980).
17. E. Fredkin, T. Toffoli, Conservative logic. *Int. J. Theor. Phys.*, vol. 21, nos. 3-4, 219-253 (1982).
18. P. Picton, Multi-valued sequential logic design using Fredkin gates. *Multiple-Valued Logic J.*, vol. 1, no. 4, 241-251 (1996).
19. J. E. Rïce, An introduction to reversible latches. *Comput. J.*, vol. 51, no. 6, 700-709 (2008).
20. H. Thapliyal, M. B. Srinivas, An extension to DNA based Fredkin gate circuits: design of reversible sequential circuits using Fredkin gates. *Proc. SPIE*, vol. 6050, 196-202 (2005).
21. M. Haghprast, K. Navi, A Novel reversible BCD adder for nanotechnology based systems. *Am. J. Appl. Sci.*, vol. 5, no. 3, 282-288 (2008).
22. M. Mohammadi, M. Eshghi, M. Haghparast, A. Bahrololoom, Design and optimization of reversible BCD Adder/Subtractor circuit for quantum and nanotechnology based systems. *World Appl. Sci. J.*, vol. 4, no. 6, 787-792 (2008).
23. G. P. Berman, G. D. Doolen, G. V. López, V. I. Tsifrinovich, A quantum full adder for a scalable nuclear spin quantum computer. *Comput. Phys. Commun.*, vol. 146, no. 3, 324-330 (2002).
24. D. I. Kamenev, G. P. Berman, R. B. Kassman, V. I. Tsifrinovich, Modeling full adder in Ising spin quantum computer with 1000 qubits using quantum maps. *Int. J. Quantum Inf.*, vol. 02, no. 03, 323-340 (2004).
25. D. Maslov, G. W. Dueck, D. Michael Miller, C. Negrevergne, Quantum circuit simplification and level compaction. *IEEE Trans. Comput-Aided Des. Integr. Circuits Syst.*, vol. 27, no. 3, 436-444 (2008).